eAnalytics - TAG 1.2

PAGE TAGGING GUIDE
## Change Log

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Comment</th>
<th>Editor</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>01.09.2011</td>
<td>final review</td>
<td>IK</td>
<td>-final-</td>
</tr>
<tr>
<td>1.1</td>
<td>09.05.2012</td>
<td>revised version</td>
<td>KS</td>
<td>-final-</td>
</tr>
<tr>
<td>1.2</td>
<td>12.05.2012</td>
<td>revised version</td>
<td>KS</td>
<td>-final-</td>
</tr>
<tr>
<td>1.3</td>
<td>18.05.2012</td>
<td>figure 1 corrected</td>
<td>KS</td>
<td>-final-</td>
</tr>
<tr>
<td>1.4</td>
<td>24.02.2013</td>
<td>eAnalytics Tag version 1.1</td>
<td>AW</td>
<td>-final-</td>
</tr>
<tr>
<td>1.5</td>
<td>10.10.2013</td>
<td>revised version</td>
<td>KS</td>
<td>-final-</td>
</tr>
<tr>
<td>1.6</td>
<td>03.01.2014</td>
<td>Changed “Open Mail” event to 19</td>
<td>KS</td>
<td>-final-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>eAnalytics Tag version 1.2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
CONTENTS

1. INTRODUCTION .................................................................................................................. 6
   1.1. Preparations .................................................................................................................. 6

2. EANALYTICS TAG STRUCTURE..................................................................................... 8

3. CODE-BLOCK 1 – PAGE LOAD TIME ............................................................................. 9

4. CODE-BLOCK 2 – VARIABLES AND JS FILE ........................................................... 10
   4.1. Tag Server .................................................................................................................. 12
   4.2. Client-ID ..................................................................................................................... 12
   4.3. Domain-ID .................................................................................................................. 12
   4.4. Online Marketing Tracking Parameter ....................................................................... 13
   4.5. Session Management ................................................................................................. 13
       4.5.1. Use your own session-ID ..................................................................................... 13
       4.5.2. Use the session-ID of the eAnalytics tag cookie: .............................................. 15
       4.5.3. Use the IP-Adresse: ......................................................................................... 15
   4.6. Cookies ...................................................................................................................... 15
   4.7. Plugin Detection ......................................................................................................... 16
   4.8. IP Mask ...................................................................................................................... 18
   4.9. eAnalytics Tag Mode ................................................................................................. 18
   4.10. Debug Mode ............................................................................................................. 19
   4.11. Non Page Requests .................................................................................................. 19
   4.12. Page Title ................................................................................................................ 21
   4.13. Page Topics ............................................................................................................. 21
   4.14. Page Attributes ....................................................................................................... 21
   4.15. URL-Parameter ........................................................................................................ 23
   4.16. Business Events ...................................................................................................... 24
4.16.1. Search ................................................................. 25
4.16.2. Order ................................................................. 26
4.16.3. Basket ................................................................. 28
4.16.4. Success ............................................................... 29
4.16.5. Login ................................................................. 30
4.16.6. Newsletter - Registration ....................................... 31
4.16.7. Service - Registration .......................................... 32
4.16.8. Product view ....................................................... 32
4.16.9. Process ............................................................... 33
4.16.10. Form Tracking .................................................... 34
4.16.11. Adspace ............................................................ 34
4.16.13. Video-Tracking ................................................... 37
4.16.14. A/B-Test ............................................................ 38
4.16.15. Viewed Site Section ............................................. 38
4.16.16. Site Search Result Click ....................................... 40
4.16.17. Click Visitor Forwarding ...................................... 41
4.16.18. Mail opened ....................................................... 42
4.16.19. Client specific events ......................................... 42
4.17. Client specific parameter .......................................... 42
4.18. Request the embedded JavaScript file ......................... 44

5. CODE-BLOCK 3 – REQUEST ......................................... 45

6. CODE-BLOCK 4 – NO JAVASCRIPT ................................. 46
   6.1. Session – ID (a) ..................................................... 46
   6.2. Domain – ID (g) .................................................... 48
   6.3. Pagetitle (e) ......................................................... 48
6.4. Page topics (έ) ....................................................................................................................... 48
6.5. JavaScript deactivated (ύ) ....................................................................................................... 48
6.6. Anonymous IP and no IP in NoScript ...................................................................................... 48

7. TRACKING OBJECTION .................................................................................................................. 49
1. INTRODUCTION

eAnalytics provides a detailed analysis of the customers’ and visitors’ behavior on your web sites. In order to achieve this goal it consists of four major components:

- eAnalytics Tag (collects data from your web sites visitors’)
- eAnalytics Data Processing (processes and loads data to the database)
- eAnalytics DB Processing (aggregates data so it can be access via the eAnalytics Portal)
- eAnalytics Portal (web based front-end that delivers dashboards, reports and other applications)

Subject to this document is the eAnalytics Tag – the page tagging solution for eAnalytics. It offers a lot of possibilities to analyze the visitors’ behavior on various types of web sites. If you want to start with only the basic functions of eAnalytics, the eAnalytics Quick-Start Guide might be helpful to give you a head start.

The eAnalytics Tag basically is a JavaScript construct that generates an invisible image request (the eAnalytics Tag) that transfers information about a visitor’s page impressions, events and sessions to the eAnalytics Tag Server. These requests are being generated by several JavaScript functions which have to be embedded in the source code of your web pages. Some JavaScript functions collect data automatically while other functions require the definition of specific values that should be transferred (like the price of a product in the shopping cart).

This document – the eAnalytics Page Tagging Guide – describes in detail and with examples, how the eAnalytics Tag needs to be integrated into your web pages.

1.1. Preparations

eAnalytics provides a JavaScript file which contains the basic functions for the eAnalytics Tag. This file is necessary in order to implement the eAnalytics Tag and it can be retrieved from www.eanalytics.de. It has to be stored on a publically accessible location on your web server – like a directory you use to store other JavaScript files already. The usage of a separate file reduces the data transfer to the visitors, since otherwise the functions have to be transferred with each request. Additionally, the integration is easier because less information needs to be put into the actual web pages and if you want to edit the JavaScript, you have to edit it once and not in every single page.

The code examples in this document include placeholders for parameter values. These placeholders can be easily identified: They start and end with a hash sign (‘#’). If you integrate the source code into your web pages you will have to replace them (including the hash-signs). This means you would replace ‘#PAGETITLE#’ with ‘Product-Detail View Soccer-Shoes (SKU 2322)’.
2. EANALYTICS TAG STRUCTURE

The following JavaScript code blocks have to be integrated into all your web pages:

1. **CODE-BLOCK 1 – Page Load Time**
   Sets the start timer for page load time and initializes variable `eat_async`

2. **CODE-BLOCK 2 – Variables and JS file**
   In this block variables are being defined and the embedded JavaScript file is being loaded

3. **CODE-BLOCK 3 – Request**
   Sets the stop timer for page load time and triggers the eAnalytics Tag

4. **CODE-BLOCK 4 – NO JAVASCRIPT**
   Triggers the eAnalytics Tag with less information in case JavaScript has been deactivated

```html
<html>
<head>
  <script>
    CODE-BLOCK 1 – Page Load Time
  </script>
</head>
<body>
  PAGE CONTENT
  
  CODE-BLOCK 2 – Variables and JS file
  CODE-BLOCK 3 – Request
  CODE-BLOCK 4 – NO JAVASCRIPT
</body>
</html>
```

*Figure 1 - eAnalytics Tag Structure*
3. CODE-BLOCK 1 – PAGE LOAD TIME

The CODE-BLOCK 1 initializes the variable eat_async in order to prepare the eAnalytics Tag to receive any information that will be processed and transferred. This is also the place where – if desired – the page load time can be determined by starting a timer. Note that this is optional.

CODE-BLOCK 1 looks like this:

```javascript
<script language="JavaScript" type="text/javascript">
    var eat_async = eat_async || [];
    eat_async.push(['eat_setPageLoadTimeStart', new Date()]);
</script>
```

Figure 2 - Code-Block 2 in detail
4. CODE-BLOCK 2 – VARIABLES AND JS FILE

The CODE-BLOCK 2 consists of two parts: First variables can be defined – secondly, the embedded JavaScript file will be requested asynchronously.

On the following pages source code examples are used to explain the functionality of the eAnalytics Tag. Before inserting the source code you will have to replace the placeholders as describe in 1.1 Preparations.

CODE-BLOCK 2 looks like this:

```javascript
<script language="JavaScript" type="text/javascript">
(function(){
  eat_async.push(['eat_setTagServerHost', '#TAGSERVERHOST#']);
  eat_async.push(['eat_setClientId', '#CLIENTID#']);
  eat_async.push(['eat_setDomainID', '#DOMAINID#']);
  eat_async.push(['eat_setSessionID', '#SESSIONID#']);
  eat_async.push(['eat_setNonPageFlag', '#YESNO#']);
  eat_async.push(['eat_setPageTitle', '#PAGETITLE#']);
  eat_async.push(['eat_setPageTopicLevel1', '#TOPIC1#']);
  eat_async.push(['eat_setPageAttribute1', '#ATTRIBUTE1#']);
  eat_async.push(['eat_setQueryParam', '#URLPARAM#']);
  eat_async.push(['eat_setEvent', '#EVENT#']);
  eat_async.push(['eat_setClientSpecificEvent', '#CLIENTSPEC#']);

  var u=(("https:" == document.location.protocol) ? "HTTPS_PATHTOFILE#" : "HTTP_PATHTOFILE#");
  var d=document, g=d.createElement('script');
  var s=d.getElementsByTagName('script')[0];
  g.type='text/javascript'; g.async=true; g.defer=true;
  g.src=u+'eat_v1_2_1.js';
  s.parentNode.insertBefore(g,s);
})();
</script>

Figure 3 - CODE-BLOCK 3 in detail

The only mandatory settings are the eAnalytics Tag Server (#TAGSERVERHOST#) and the path to the embedded JavaScript file.

The embedded JavaScript file (eat_v1_2_1.js) contains two global parameter for the usage of the eAnalytics Tags. With these global parameters you can call functions to set specific internal parameters and trigger the eAnalytics Tag request. The internal parameters have default settings, so in most cases these settings will suffice and do not need to be changed. Not all possible settings are mentioned in the figure above. But on the following pages all of these settings will be described in detail.
The eAnalytics Tag offers a wide range of settings. Although it is not necessary to set them, eAnalytics recommends their definition in order to optimize the value and the readability of the reports.
4.1. Tag Server

The eAnalytics Tag Server can be set using the following JavaScript call:

```javascript
eat_async.push(['eat_setTagServerHost', '#TAGSERVERHOST#']);
```

`#TAGSERVERHOST#` has to be replaced by the name or IP address of your eAnalytics Tag Server (like 'my-domain.de' or '164.22.23.71' without double quotes) - usually the server you installed eAnalytics on. This setting is not optional.

4.2. Client-ID

The placeholder `#CLIENTID#` represents the client. Usually the default setting will suffice. It is set to ‘1000’ which is the default client ID that has been specified within the eAnalytics Server installation.

4.3. Domain-ID

The placeholder `#DOMAINID#` represents the domain that will be reported in the eAnalytics Portal and usually it represents one physical domain (like www.eanalytics.de).

If you want to analyze only one domain, the default setting will suffice. It is set to ‘100’ which is the default domain-ID that has been specified within the eAnalytics Server installation.

If you want to analyze more than one domain you can set the domain-ID with this JavaScript call:

```javascript
eat_async.push(['eat_setDomainID', #DOMAINID#]);
```

In this case you would also have to add the other domains into the corresponding database table: db.domain.

This can be done by connecting to your eAnalytics database and execute the following statement:

```
INSERT INTO db.domain( domain_id,domain_name,data_to_db_fl,data_to_dm_fl, entailed_visit_days_1,entailed_visit_days_2, entailed_visit_days_3,entailed_visit_days_max) VALUES( #DOMAINID#, #DOMAINNAME#,1,1,0,5,10,30);
```

Example:

```
INSERT INTO db.domain( domain_id,domain_name,data_to_db_fl,data_to_dm_fl, entailed_visit_days_1,entailed_visit_days_2, entailed_visit_days_3,entailed_visit_days_max) VALUES( #DOMAINID#, #DOMAINNAME#,1,1,0,5,10,30);
```

Although it is possible to summarize various physical domains into one domain-ID as long as a visitor can be tracked with the same session-ID.
entailed_visit_days_1,entailed_visit_days_2, entailed_visit_days_3,entailed_visit_days_max) VALUES( 201,'mydomain.de',1,1,0,5,10,30);

4.4. Online Marketing Tracking Parameter

The online marketing parameters in the JavaScript are set by default to the following types:

```javascript
var eat_source = 'utm_source|eatso';
var eat_campaign = 'utm_campaign|eatca';
var eat_content = 'utm_content|eatco';
var eat_channel = 'utm_channel|eatch';
var eat_term = 'utm_term|eatte';
var eat_recipient = 'eatre';
```

With these default settings Google Analytics parameters are automatically tracked. Usually this default settings suffice. If you want to use different parameter names, you have to change the parameter settings.

This could be done as follows:

```javascript
eat_async.push(['eat_setSource', '#SOURCE#']);
eat_async.push(['eat_setCampaign', '#CAMPAIGN#']);
eat_async.push(['eat_setContent', '#CONTENT#']);
eat_async.push(['eat_setChannel', '#CHANNEL#']);
eat_async.push(['eat_setTerm', '#TERM#']);
eat_async.push(['eat_setRecipient', '#RECIPIENT#']);
```

The eAnalytics Link Tagging Guide describes these parameters in details and guides you to use them.

4.5. Session Management

The eAnalytics Tag offers three option of reconstructing a visit:

4.5.1. Use your own session-ID

You can use the session-ID that has already been generated by your web site. The maximum length is 250 characters:

```javascript
eat_async.push(['eat_setSessionID', '#SESSIONID#']);
```
e.g.:

eat_async.push(['eat_setSessionID', '342345jh234ljh2qr2344fsadf']);
4.5.2. Use the session-ID of the eAnalytics tag cookie:

If there is no session-ID set, the eAnalytics Tag will use generate a session-ID and store it in a cookie. This is only possible when the parameter `eat_cookieStatus` is set to ‘temp’ or ‘full’. The default setting in the JavaScript-File is ‘full’.

If you want to use our default cookie to track your visitors – you can just leave everything to default.

4.5.3. Use the IP-Adresse:

If the parameter `eat_sessionID` has not been set and `eat_cookieStatus` is set to ‘off’ or the visitor will not accept cookies, eAnalytics uses the IP – address and some technical info of the visitor to build a session. This procedure is very imprecise.

The default timeout for a session is 120 minutes.

4.6. Cookies

The parameter `eat_cookieStatus` sets the type of the cookie for session reconstruction and for tracking recurring visitors. The following three values are possible:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>temp</td>
<td>activate session cookies, activate not persistent cookies for recurring visitors identification</td>
</tr>
<tr>
<td>full</td>
<td>activate session cookies and persistent cookies for recurring visitors identification</td>
</tr>
<tr>
<td>off</td>
<td>deactivate all cookies</td>
</tr>
</tbody>
</table>

I you want to change the parameter the following could be used:

```javascript
    eat_async.push(['eat_setCookieStatus', '#COOKIE_STATUS#']);
```

If the parameter `eat_sessionID` is set the value of the session cookie will be overwritten. If the parameter `eat_cookieStatus` is set to the values ‘temp’ or ‘full’, the parameter `eat_sessionID` should not be set.
If it is not possible to set the Session-ID, the usage of cookies is strongly recommended. Without cookies the session reconstruction will be based on IP address, technical information and the time, the reconstruction will then be imprecise.

The session cookie will be set with a lifetime of 120 minutes and the persistent cookie with a lifetime of one year. These values can be changed individually.

The session reconstruction can also be handled by a 3rd Party Cookie. Although 3rd Party Cookies might be blocked by some visitor’s browsers, 3rd Party Cookies might be a good solution if you want to summarize more than one domain into one domain-ID in order to track the visitors’ behavior over all of your websites.

The default value in the JavaScript-File is ‘off’. To activate the 3rd-Party cookie the value has to be set to:

```javascript
eat_async.push(['eat_setThirdParty', 'on']);
```

Is the parameter set to ‘on’, the tag request is redirected to a PHP-Script. The PHP-Script modifies the tag request and sends it to the tag server.

Please note that you will have to adjust the hostname of your eAnalytics Tag Server in the variable $adressstring and the function setcookie() in the PHP files: eat0.php and eat1.php.

### 4.7. Plugin Detection

The parameter `eat_pluginStatus` controls the detection of plugins in the client browser of your visitors. The possible values are ‘select’, ‘full’ or ‘off’:

<table>
<thead>
<tr>
<th>select</th>
<th>Needs information which plugins have to be identified. Currently only the Flash Player and Windows Media Player are available. They have to be activated by the corresponding parameters:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>`(eat_pluginFlash = true</td>
</tr>
<tr>
<td>full</td>
<td>Activates the detection of all Plugins</td>
</tr>
<tr>
<td>off</td>
<td>Deactivates the detection of all Plugins</td>
</tr>
</tbody>
</table>

If you want to change the parameter the following could be used:

```javascript
eat_async.push(['eat_setPluginStatus', '#PLUGIN_STATUS#']);
```
The default value in the JavaScript-File is set to ‘full’. If the plugins in the client browser should not identify the value should be set to:

```javascript
eat_async.push(['eat_setPluginStatus', 'off']);
```
If the value is set to ‘select’ the default detection plugins are:

```javascript
eat_pluginFlash = true
eat_pluginWMedia = false
```

In order to change these setting you can use:

```javascript
eat_async.push(['eat_setPluginWMedia', #TRUEorFALSE#]);
eat_async.push(['eat_setPluginFlash', #TRUEorFALSE#]);
```

e.g: `eat_async.push(['eat_setPluginWMedia', true]);`
```javascript
eat_async.push(['eat_setPluginFlash', false]);
```

4.8. IP Mask

The parameter `eat_ipMask` activates one of eAnalytics data privacy features. If it is set to ‘1’ a masked version\(^2\) of the visitors IP address will be sent to the eAnalytics Tag Server. This is the default setting.

```javascript
var eat_ipMask = 1;    // 1= anonymous IP  0= No IP
```

If you want to change the parameter the following could be used:

```javascript
eat_async.push(['eat_setIPMask', #IPMASK#]);
```

If you want to disable the logging of the IP address completely, you would need to set the value to ‘0’. Please note that in this case the location of visitors by countries, regions or cities will not be possible. The IP address (full or masked) will in no case be saved in the eAnalytics database.

4.9. eAnalytics Tag Mode

The eAnalytics Tag can be turned off using the parameter `eat_eatStatus`.

<table>
<thead>
<tr>
<th>request</th>
<th>eAnalytics Tag operates</th>
</tr>
</thead>
<tbody>
<tr>
<td>off</td>
<td>eAnalytics Tag is turned off</td>
</tr>
</tbody>
</table>

\(^2\) The last digits of the IP address will be set to 0. “78.46.198.252” will be shortened to “78.46.198.0”.

18 Integrated Analytics GmbH > www.eAnalytics.de
The default value in the JavaScript-File is set to ‘request’ If the eAnalytics Tag requests should be deactivated, the parameter has to be set to ‘off’:

    eat_async.push(['eat_setEatStatus', 'off']);

Setting the parameter in CODE-BLOCK 2 can turn the eAnalytics Tag off for certain pages.

### 4.10. Debug Mode

The parameter `eat_debug` offers a debugging mechanism for the eAnalytics Tag. The default value of the parameter is ‘0’:

```javascript
var eat_debug = 0     // 0= no debug
```

In this case the debug mode is deactivated.

Setting the parameter value to ‘1’:

```javascript
var eat_debug = 1;    // 1= debug (tag request deactivated)
```

No request is send to the tag server. A JavaScript pop-up (alert) will show all parameter with their values to verify the tag in a development environment.

Setting the parameter value to ‘2’:

```javascript
var eat_debug = 2;   // 2= debug (tag request activated)
```

The request is send to the tag server and a JavaScript pop-up (alert) will show all parameter values.

If you want to change the parameter the following call could be used:

```javascript
eat_async.push(['eat_setDebug', #DEBUG#]);
```

### 4.11. Non Page Requests

This parameter can be used to mark web pages that should not be considered as ‘page views’, however, provide important information. This may be necessary if links from an online partner lead to a certain page and redirect visitors automatically to another web page. In this case, the request of the first page would identify the source of the session, while the request itself should not be counted as a ‘page view’ as it had been generated automatically.
The default value in the JavaScript-File is ‘no’. Otherwise you will have to add the following to CODE-BLOCK 2 of the corresponding page:

```javascript
eat_async.push(['eat_setNonPageFlag','yes']);
```
4.12. Page Title

The parameter `eat_pageTitle` can be used to transfer the title of a web page:

Example: `eat_async.push(['eat_setPageTitle', 'Homepage']);`

The length of the page title is restricted to 250 characters.

If this parameter will not be defined, the page title from the HTML `<title>` will be sent instead. This will be the title of the web page in the eAnalytics portal.

4.13. Page Topics

With these parameters it is possible to assign up to three topics for each web site:

```javascript
eat_async.push(['eat_setPageTopicLevel1', '#TOPIC1#']);
eat_async.push(['eat_setPageTopicLevel2', '#TOPIC2#']);
eat_async.push(['eat_setPageTopicLevel3', '#TOPIC3#']);
```

Example: `eat_async.push(['eat_setPageTopicLevel1', 'Shop do it yourself']);`

```javascript
eat_async.push(['eat_setPageTopicLevel2', 'electronic products']);
eat_async.push(['eat_setPageTopicLevel3', 'cutouts']);
```

Each of these three parameters can contain a string with a maximum of 250 characters. This topic classification is offered in the report. This way a page analysis a certain topic level can be achieved.

4.14. Page Attributes

Within these parameters you assign up to three custom attributes for each web page. This might be useful if you want to track web page attributes that are no page topics (like the author of an article):

```javascript
eat_async.push(['eat_setPageAttribute1', '#ATTRIBUTE1#']);
eat_async.push(['eat_setPageAttribute2', '#ATTRIBUTE1#']);
eat_async.push(['eat_setPageAttribute3', '#ATTRIBUTE1#']);
```

EXAMPLE: `eat_async.push(['eat_setPageAttribute1', 'Shop']);`

```javascript
eat_async.push(['eat_setPageAttribute2', 'Products']);
eat_async.push(['eat_setPageAttribute3', 'Detail-Product-View']);
```
Each of these three parameters can contain a string with a maximum of 250 characters. In order to analyze these values you can select these attributes in the web page reports in the eAnalytics Portal.
4.15. URL-Parameter

The eAnalytics Tag transfers only the requested web-resources (e.g. ‘pview.php’) and not the whole request.

http://www.myshop.de/pview.php?prodSKU=123&sid=HISfyzTBOzC&t=2

Only the name of the web-resource (‘pview.php’) is submitted to the database. That means a detail analysis for the request ‘pview.php’ is not possible (e.g. ‘How many request exist for the ProductSKU?’).

If an analysis of the dynamic page parameter is necessary, every parameter has to be defined as an eat_QueryParameter. If - as in the example above - the query parameters ‘prodSKU’ and ‘t’ are necessary for a detailed analysis, the parameters have to be defined like this:

```javascript
eat_async.push(['eat_addQueryParameter', 'prodSKU']);
eat_async.push(['eat_addQueryParameter', 't']);
```

or shorter: `eat_async.push(['eat_addQueryParameter', 'prodSKU','t']);`

In this case the query string parameter ‘sid’ would not be transferred.
4.16. Business Events

The parameter `eat_event` describes business related events on the web site, such as a product detail view or an order confirmation.

The following events can be specified:

<table>
<thead>
<tr>
<th>Event-ID</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Search</td>
</tr>
<tr>
<td>2</td>
<td>Order</td>
</tr>
<tr>
<td>3</td>
<td>Basket</td>
</tr>
<tr>
<td>4</td>
<td>Success</td>
</tr>
<tr>
<td>5</td>
<td>Login</td>
</tr>
<tr>
<td>6</td>
<td>Newsletter Registration</td>
</tr>
<tr>
<td>7</td>
<td>Service Registration</td>
</tr>
<tr>
<td>8</td>
<td>Product View</td>
</tr>
<tr>
<td>9</td>
<td>Process</td>
</tr>
<tr>
<td>10</td>
<td>Form Tracking</td>
</tr>
<tr>
<td>11</td>
<td>Adspace</td>
</tr>
<tr>
<td>12</td>
<td>http-Status Code</td>
</tr>
<tr>
<td>13</td>
<td>Video Tracking</td>
</tr>
<tr>
<td>14</td>
<td>AB-Test</td>
</tr>
<tr>
<td>15</td>
<td>Preferred Category</td>
</tr>
<tr>
<td>16</td>
<td>Site Search Result Click</td>
</tr>
<tr>
<td>17</td>
<td>Click Visitor Forwarding</td>
</tr>
<tr>
<td>18</td>
<td>Mails opened</td>
</tr>
</tbody>
</table>

Each event can include certain information. In this section you will find an explanation and examples how to do this.

The first information is always the event-ID. It is possible to define up to 20 events on a single web page. For example - if a client logs in during the order process, both events – login (event-ID 5) and the order (event-ID 2) can be set in the web page:

```javascript
eat_async.push(['eat_addEvent',5,1,'aaron','kd32765']);
eat_async.push(['eat_addEvent',2,1,'od32433', 'wk23398', 'kd3275', '19.99', '16.88', 'EUR']);
```

If individual values cannot be set, the field can remain blank(`"`) even for Integer or Decimal values. The number and order of parameters for each event has to stay the same.
4.16.1. Search

For the search event the following information can be set:

<table>
<thead>
<tr>
<th>Information</th>
<th>Type</th>
<th>Required</th>
<th>Max Length</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search string</td>
<td>String</td>
<td>yes</td>
<td>250</td>
<td></td>
</tr>
<tr>
<td>Number of search results</td>
<td>Integer</td>
<td>yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Search type</td>
<td>Integer</td>
<td></td>
<td>Values</td>
<td>Meaning</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>others</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>One word search</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>Multiple word search</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>Article No. search</td>
</tr>
<tr>
<td>Result page</td>
<td>Integer</td>
<td>yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Max. shown result position</td>
<td>Integer</td>
<td>yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Search category</td>
<td>String</td>
<td></td>
<td>250</td>
<td></td>
</tr>
<tr>
<td>Search sort attribute</td>
<td>String</td>
<td></td>
<td>250</td>
<td></td>
</tr>
<tr>
<td>Search sort direction</td>
<td>String</td>
<td></td>
<td>250</td>
<td></td>
</tr>
</tbody>
</table>

Notation:

```javascript
eat_async.push([’eat_addEvent’, #EVENTID#, #SEARCHSTRING#, #RESULTS#, #SEARCHTYPE#, #RESULTPAGE#, #MAXRESULTPOSITION#, #SEARCHCATEGORY#, #SEARCHSORT#, #SEARCHSORTDIRECTION#]);

e.g.: eat_async.push([’eat_addEvent’, 1,’Shoe’,50,1,1,20,’Fashion’,’Price’, ’descending’]);
```
4.16.2. Order

For the order event the following information can be set:

<table>
<thead>
<tr>
<th>Information</th>
<th>Type</th>
<th>Required</th>
<th>Max Length</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>Integer</td>
<td>yes</td>
<td></td>
<td>Values</td>
</tr>
<tr>
<td>Order No.</td>
<td>String</td>
<td>yes</td>
<td>250</td>
<td>Meaning</td>
</tr>
<tr>
<td>Basket ID</td>
<td>String</td>
<td></td>
<td>250</td>
<td></td>
</tr>
<tr>
<td>Client ID</td>
<td>String</td>
<td></td>
<td>250</td>
<td></td>
</tr>
<tr>
<td>Revenue</td>
<td>Decimal</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revenue Net</td>
<td>Decimal</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Currency</td>
<td>String</td>
<td></td>
<td>250</td>
<td>Alpha ISO-Code currency</td>
</tr>
<tr>
<td>Shipment</td>
<td>Decimal</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Discount</td>
<td>Decimal</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Notation:

eat_async.push(['eat_addEvent', '#EVENTID#,#ACTION#,#ORDENO#,#BASKETID#', '#CLIENTID#,#REVENUE#,#REVENUENET#,#CURRENCY#,#SHIPMENT#,#DISCOUNT#]);

e.g.: eat_async.push(['eat_addEvent', '2,1','od324323','wk234398', 'kd32765',19.99,16.88,'EUR',3.90,4.23]);

3 Order can only be labeled as canceled within a period of 24h.
4.16.3. Basket

For the basket event the following information can be set:

<table>
<thead>
<tr>
<th>Information</th>
<th>Type</th>
<th>Required</th>
<th>Max Length</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basket ID</td>
<td>String</td>
<td>yes</td>
<td>250</td>
<td></td>
</tr>
<tr>
<td>Action</td>
<td>Integer</td>
<td>yes</td>
<td></td>
<td>Values</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>Product ID</td>
<td>String</td>
<td>yes</td>
<td>250</td>
<td></td>
</tr>
<tr>
<td>Product name</td>
<td>String</td>
<td></td>
<td>250</td>
<td></td>
</tr>
<tr>
<td>Product category1</td>
<td>String</td>
<td></td>
<td>250</td>
<td></td>
</tr>
<tr>
<td>Product category2</td>
<td>String</td>
<td></td>
<td>250</td>
<td></td>
</tr>
<tr>
<td>Product category3</td>
<td>String</td>
<td></td>
<td>250</td>
<td></td>
</tr>
<tr>
<td>Quantity</td>
<td>Integer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Price per unit</td>
<td>Decimal</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Price per unit Net</td>
<td>Decimal</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Currency</td>
<td>String</td>
<td></td>
<td>250</td>
<td>Alpha ISO-Code currency</td>
</tr>
</tbody>
</table>

Notation:

eat_async.push(['eat_addEvent',
                 #EVENTID#,'#BASKETID#', #ACTION#, #PRODUCTID#,
                 '#PRODUCTNAME#', '#PRODUCTCATEGORY1#', '#PRODUCTCATEGORY2#',
                 '#PRODUCTCATEGORY3#', #QUANTITY#, #PRICEPERUNIT#,
                 #PRICEPERUNITNET#,'#CURRENCY#']);

e.g.: eat_async.push(['eat_addEvent', 3, 'wk234398', 1, 23, 'Sport Shoe Extreme', 'Fashion', 'Shoes', 'Sports', 2, 2.99, 2.77, 'EUR']);
4.16.4. Success

Depending on the business model, events like a download of a PDF file or sending a form might represent a website success. Such success events can be defined like this:

<table>
<thead>
<tr>
<th>Information</th>
<th>Type</th>
<th>Required</th>
<th>Max Length</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lead Type</td>
<td>String</td>
<td>yes</td>
<td>250</td>
<td>Name of Success</td>
</tr>
<tr>
<td>Lead Type Category</td>
<td>String</td>
<td></td>
<td>250</td>
<td>Success class</td>
</tr>
</tbody>
</table>

Notation:

```javascript
eat_async.push(['eat_addEvent', #EVENTID#, '#LEADTYPE#', '#LEADTYPECATEGORY#']);
e.g.: eat_async.push(['eat_addEvent', 4, 'Download Contract', 'Downloads']);
```

Additionally it is possible to send a success event calling:

```javascript
eat_async.push(['eat_ClickSuccessEvent', '#LEADTYPE#', '#LEADTYPECATEGORY#']);
```

This might be helpful if a success is a click on a link:

```javascript
<a href='download_information.pdf' onclick='return eat_async.push(['eat_ClickSuccessEvent', 'Download contract', 'Download']);'>
Link to PDF</a>
```
4.16.5. Login

For the login event the following information can be set:

<table>
<thead>
<tr>
<th>Information</th>
<th>Type</th>
<th>Required</th>
<th>Max Length</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>Integer</td>
<td>yes</td>
<td></td>
<td>1 – Login / 2 – Logoff</td>
</tr>
<tr>
<td>Login name</td>
<td>String</td>
<td></td>
<td>250</td>
<td></td>
</tr>
<tr>
<td>Client ID</td>
<td>String</td>
<td></td>
<td>250</td>
<td></td>
</tr>
</tbody>
</table>

Notation:

```javascript
eat_async.push(['eat_addEvent',
  #EVENTID#,#ACTION#,'#LOGIN#','#CLIENTID#']);
e.g.: eat_async.push(['eat_addEvent', 5,1,'eatlog', 'kd32765']);
```

On most cases the login is done by the client’s email address. In this case replace #CLIENTID# with the email address.
4.16.6. Newsletter - Registration

For the Newsletter – Registration event the following information can be set:

<table>
<thead>
<tr>
<th>Information</th>
<th>Type</th>
<th>Required</th>
<th>Max Length</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client ID⁴</td>
<td>String</td>
<td></td>
<td>250</td>
<td></td>
</tr>
<tr>
<td>Action</td>
<td>Integer</td>
<td>yes</td>
<td></td>
<td>Values</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>Newsletter ID</td>
<td>String</td>
<td></td>
<td>250</td>
<td></td>
</tr>
<tr>
<td>Newsletter name</td>
<td>String</td>
<td></td>
<td>250</td>
<td></td>
</tr>
</tbody>
</table>

Notation:

```javascript
eat_async.push(['eat_addEvent',
  '#EVENTID#','#CLIENTID#','#ACTION#','#NEWSLETTERID#','#NEWSLETTERNAME#']);

e.g.: eat_async.push(['eat_addEvent',6,'kd32765',1,'nl32','Discounts']);
```

⁴ On most cases the registration is done by the client email address. In this case replace #CLIENTID# with the email address.
4.16.7. Service - Registration

For the Service – Registration following information can be set:

<table>
<thead>
<tr>
<th>Information</th>
<th>Type</th>
<th>Required</th>
<th>Max Length</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClientID</td>
<td>String</td>
<td></td>
<td>250</td>
<td></td>
</tr>
<tr>
<td>Action</td>
<td>Integer</td>
<td>yes</td>
<td></td>
<td>Values</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Meaning</td>
</tr>
<tr>
<td>ServiceID</td>
<td>String</td>
<td></td>
<td>250</td>
<td></td>
</tr>
<tr>
<td>ServiceName</td>
<td>String</td>
<td></td>
<td>250</td>
<td></td>
</tr>
</tbody>
</table>

Values
- 1  Service subscript
- 2  Services view
- 3  Service unsubscript

Notation:

eat_async.push(['eat_addEvent', '#EVENTID', '#CLIENTID', '#ACTION', '#SERVICEID', '#SERVICENAME']);
e.g.: eat_async.push(['eat_addEvent', 7, 'kd32765', 1, 'servtrack', 'Shipment tracking']);

4.16.8. Product view

For the product view event the following information can be set:

<table>
<thead>
<tr>
<th>Information</th>
<th>Type</th>
<th>Required</th>
<th>Max Length</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product ID</td>
<td>String</td>
<td>yes</td>
<td>250</td>
<td></td>
</tr>
<tr>
<td>Product name</td>
<td>String</td>
<td></td>
<td>250</td>
<td></td>
</tr>
<tr>
<td>Product category1</td>
<td>String</td>
<td></td>
<td>250</td>
<td></td>
</tr>
<tr>
<td>Product category2</td>
<td>String</td>
<td></td>
<td>250</td>
<td></td>
</tr>
<tr>
<td>Product category3</td>
<td>String</td>
<td></td>
<td>250</td>
<td></td>
</tr>
<tr>
<td>View type</td>
<td>String</td>
<td></td>
<td>250</td>
<td></td>
</tr>
</tbody>
</table>

Notation:

eat_async.push(['eat_addEvent', '#EVENTID', '#PRODUCTID', '#PRODUCTNAME', '#PRODUCTCATEGORY1', '#PRODUCTCATEGORY2', '#PRODUCTCATEGORY3', '#VIEWTYPE']);
e.g.: eat_async.push(['eat_addEvent', 8, 23, 'RUNNING Shoe XC', 'Fashion', 'Sports', 'Shoe', 'Productvideo']);
If preferably all fields are filled for this event the readability of the product reports will optimal.

### 4.16.9. Process

Processes on web sites can be defined with this event. Every steps of the process has to be implemented in the corresponding web pages. The following information should be set:

<table>
<thead>
<tr>
<th>Information</th>
<th>Type</th>
<th>Required</th>
<th>Max Length</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process ID</td>
<td>Integer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Process name</td>
<td>String</td>
<td></td>
<td>250</td>
<td></td>
</tr>
<tr>
<td>Process step ID</td>
<td>Integer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Process step name</td>
<td>String</td>
<td></td>
<td>250</td>
<td></td>
</tr>
</tbody>
</table>

**Notation:**

```javascript
eat_async.push(['eat_addEvent', #EVENTID#, #PROCESSID#, #PROCESSNAME#, #PROCESSSTEPID#, #PROCESSSTEPNAME#]);
```

e.g.: `eat_async.push(['eat_addEvent', 9, 1, 'Order', 3, 'Order Confirmation']);`

The analysis of a process is possible when all web pages of a certain process have been equipped with this event.

The notation example shows the process ‘Order’ with the process-ID ‘1’ and the process step-ID ‘3’ ‘Order Confirmation’. Earlier steps in such a process ‘Order’ might be process step-ID ‘1’ for ‘Address Information” and process step-ID ‘2’ for ‘Payment’.

It is recommended to define the process step names to get readable reports.

Another example for a process might be the newsletter registration. In this case, because process-ID ‘1’ is already used for ‘Order’, process-ID ‘2’ will be used to identify all process steps of the newsletter registration.
### 4.16.10. Form Tracking

The form tracking event shows which fields are filled out by the user and which are not. The special tracking function `eat_setFormTrackingEvent` needs to be set within ‘onChange’ of all fields in the form:

<table>
<thead>
<tr>
<th>Information</th>
<th>Type</th>
<th>Required</th>
<th>Max Length</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form name</td>
<td>String</td>
<td>yes</td>
<td>250</td>
<td>Name of form</td>
</tr>
<tr>
<td>Form step ID</td>
<td>Integer</td>
<td>yes</td>
<td></td>
<td>Position of field</td>
</tr>
<tr>
<td>Form step</td>
<td>String</td>
<td>yes</td>
<td>250</td>
<td>Name of field</td>
</tr>
<tr>
<td>Form step type</td>
<td>Integer</td>
<td></td>
<td></td>
<td>Values</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

Notation:

```javascript
eat_async.push(['eat_setFormTrackingEvent','#FORMNAME#','#FORMSTEPID#','#FORMSTEP#','#FORMSTEPTYPE#']);
```

e.g.:

```html
<input name='firstname' onChange="eat_async.push(['eat_setFormTrackingEvent','Reg_Form',1,'Firstname',1]);" type='text'>
```

### 4.16.11. Adspace

The efficiency of internal advertising spaces can be tracked with the event ‘Adspace’. The event consists of two parts: one for impression of an adspace and one for a click it.

1. Medium Impression
2. Medium Click

The ‘Medium Impression’ should send the following information:

<table>
<thead>
<tr>
<th>Information</th>
<th>Type</th>
<th>Required</th>
<th>Max Length</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>Integer</td>
<td>yes</td>
<td></td>
<td>1 = Medium Impression</td>
</tr>
<tr>
<td>Adspace</td>
<td>String</td>
<td>yes</td>
<td>per Field 64</td>
<td></td>
</tr>
</tbody>
</table>

Notation:

```javascript
eat_async.push(['eat_addEvent', '#EVENTID#','ACTION#','#ADSPACE#']);
```
The ‘Medium Click’ is set in the link of the medium with the parameter ‘eat_adspace’:

Notation:

```
http://www.myshop.de/?eat_adspace=NAME_POSITION_STYLE_CLASS_product_PRODUCTID
```

e.g.:

```
http://www.eAnalytics.de/shop/books/detailview.php?product_id=23562345
&eat_adspace=DiscountBooks_top%20left_Bold%20Headline_Book%20Teaser_product_23562345
```

For the implementation of the ‘Medium Impression’ and the ‘Medium Click’ the following notation of the #ADSPACE# is **mandatory**:

**NAME_POSITION_STYLE_CLASS_product_PRODUCTID**

<table>
<thead>
<tr>
<th>Information</th>
<th>Type</th>
<th>Required</th>
<th>Max Length</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>String</td>
<td>yes</td>
<td>64</td>
<td>DiscountBooks</td>
</tr>
<tr>
<td>Position</td>
<td>String</td>
<td></td>
<td>64</td>
<td>TopLeft</td>
</tr>
<tr>
<td>Style</td>
<td>String</td>
<td></td>
<td>64</td>
<td>BoldHeadline</td>
</tr>
<tr>
<td>Class</td>
<td>String</td>
<td></td>
<td>64</td>
<td>Teaser</td>
</tr>
<tr>
<td>Type</td>
<td>String</td>
<td></td>
<td>64</td>
<td>‘product’ for the optional field</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Product-ID or ‘DiscountCampain”</td>
</tr>
<tr>
<td>Product-ID</td>
<td>String</td>
<td></td>
<td>64</td>
<td>Required if type = product</td>
</tr>
</tbody>
</table>

The last field ‘Product-ID’ is **only** scanned by the eAnalytics Tag if the field ‘Type’ is set to ‘product’. The character ‘_’ is used as the field delimiter and therefore restricts the fields to not have this character in them. Furthermore the fields need to be URL-encoded. In the example above spaces are encoded with ‘%20’.

Fields with no defined content can be left empty but it is necessary that the minimum of 4 ‘_’ are set, so that the event can be triggered.

If the parameter ‘eat_adspace’ is set and the page triggers more than one request of the eAnalytics (e.g. caused by ajax requests) this event would be send multiple times. In order to avoid this, it is possible to deactivate the automatic event generation with the parameter ‘eat_ignEvent’. This parameter can have the values ‘no’ and ‘yes’. The default value is ‘no’,
that means the event ‘Adspace’ is sent each time the tag request is triggered. To change the value, you can use:

```javascript
eat_async.push(['eat_set_ignoreEvent', 'yes']);
```

This event track all errors and can be integrated into the corresponding error pages (e.g. 404 page not found).

<table>
<thead>
<tr>
<th>Information</th>
<th>Type</th>
<th>Required</th>
<th>Max Length</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>http-Status Code</td>
<td>Integer</td>
<td>yes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Notation:

```javascript
eat_async.push(['eat_addEvent', #EVENTID#, #http-status code#]);
```
e.g.: `eat_async.push(['eat_addEvent', 12, 404]);`

---

4.16.13. Video-Tracking

For the video tracking event the following information can be relayed:

<table>
<thead>
<tr>
<th>Information</th>
<th>Type</th>
<th>Required</th>
<th>Max Length</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video state</td>
<td>Integer</td>
<td></td>
<td></td>
<td>Values Meaning</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 Start</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2 Stop</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3 Still Playing</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4 Jumping</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5 Fullscreen</td>
</tr>
<tr>
<td>Video duration in seconds</td>
<td>Integer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Video name</td>
<td>String</td>
<td></td>
<td>250</td>
<td></td>
</tr>
<tr>
<td>Video type</td>
<td>String</td>
<td></td>
<td>250</td>
<td></td>
</tr>
</tbody>
</table>

Notation:

```javascript
eat_async.push(['eat_addEvent', #EVENTID#, #VIDEOSTATE#, #VIDEODURATION#, #VIDEONAME#, #VIDEOTYPE#]);
```
e.g.: `eat_async.push(['eat_addEvent', 13,1,24323, 'Daily News', 'Newsvideo']);`

Note for the video duration depending on the video state:

- 1=Start: sum duration

Using the A/B-Test event you can identify the success of alternative designs and content. This event can either be implemented by the default event notation or using the query string parameter eat_ABTest:

<table>
<thead>
<tr>
<th>Information</th>
<th>Type</th>
<th>Required</th>
<th>Max Length</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test name</td>
<td>String</td>
<td>yes</td>
<td>250</td>
<td></td>
</tr>
<tr>
<td>Test value</td>
<td>String</td>
<td>yes</td>
<td>250</td>
<td></td>
</tr>
</tbody>
</table>

Notation Event JavaScript:

```javascript
eat_async.push(['eat_addEvent', #EVENTID#, '#TESTNAME', '#TESTVALUE']);
e.g.: eat_async.push(['eat_addEvent', 14, 'AlternativeB', 'withBox']);
```

Notation URL-Parameter:

```
http://www.meinshop.de/?eat_ABTest=TESTNAME_TESTVALUE
e.g. http://www.myshop.de/?eat_ABTest=AlternativeB_withBox
```

If the parameter ‘eat_ABTest’ is set and the page triggers more than one request of the eAnalytics (e.g. caused by ajax requests) this event would be send multiple times. In order to avoid this, it is possible to deactivate the automatic event generation with the parameter ‘eat_ignEvent’. This parameter can have the values ‘no’ and ‘yes’. The default value is ‘no’, that means the event ‘A/B-Test’ is sent each time the tag request is triggered. To change the value, you can use:

```javascript
eat_async.push(['eat_setIgnoreEvent', 'yes']);
```

4.16.15. Viewed Site Section

With the event ‘Viewed Site Section’ it is possible to track the affinity of visitors to certain parts of your web sites like certain product categories. In the parameter ‘Section Priority’ you can set the quality rating:

```
http://www.meinshop.de/?eat_ABTest=TESTNAME_TESTVALUE
```
<table>
<thead>
<tr>
<th>Information</th>
<th>Type</th>
<th>Required</th>
<th>Max Length</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Viewed Site Section</td>
<td>String</td>
<td>yes</td>
<td>250</td>
<td></td>
</tr>
<tr>
<td>Section Priority</td>
<td>Integer</td>
<td>yes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Notation:

```javascript
eat_async.push(['eat_addEvent', #EVENTID#, #VIEWEDSITESECTION#, #SECTIONPRIORITY#]);

e.g.: eat_async.push(['eat_addEvent', 15, 'Fashion', 32]);
```

This event is the base for the attribute ‘preferred section’. Based on this event, for every ‘viewed section’ and every visit the highest value will calculated. If one user visit in one session visits the section ‘jewelry’ twice with a value of 20 and once the section ‘fashion’ with a value of 50 than the preferred section is ‘fashion’ (because: ‘jewelry’ = 40 & ‘fashion’ = 50).

In an e-commerce shop other values like product views, baskets or orders influence the calculation of this event. Every event is assigned to the ‘viewed site section’ until an event with a different ‘viewed site section’ occurs. The values of the additional events for the calculation are set in the corresponding control table (db.event_type). The following default values are set by default:

- Product view → 10
- Basket → 50
- Order → 200

4.16.16. Site Search Result Click

This event helps to analyze the result of a search query. The parameter `eat_sitesearch` has to be implemented on every result link on the result pages.

Notation:

```
http://www.myshop.de/?eat_sitesearch=SEARCHTERM_RESULTPOSITION_RESULTDESCRIPTION
```

The following implementation is mandatory:

```
SEARCHTERM = Search term
RESULTPOSITION = Position of the result on the search result page
RESULTDESCRIPTION = Description of the search result
```

The character ‘_’ is used as the field delimiter that means the defined content do not contained the character ‘_’. Fields with no defined content can be empty, but it is necessary that 2 ‘_’ are set; otherwise the event will not be send.
If the parameter ‘eat_sitesearch’ is set and the page triggers more than one request of the eAnalytics (e.g. caused by ajax requests) this event would be send multiple times. In order to avoid this, it is possible to deactivate the automatic event generation with the parameter ‘eat_ignEvent’. This parameter can have the values ‘no’ and ‘yes’. The default value is ‘no’, that means the event ‘Site Search Result Click’ is sent each time the tag request is triggered. To change the value, you can use:

```javascript
eat_async.push(['eat_set_ignoreEvent', 'yes']);
```

### 4.16.17. Click Visitor Forwarding

The event ‘Click Visitor Forwarding’ tracks the exit of the visitor when an external link is used. The function `eat_VisitorForwarding` has to be implemented in `onClick` of the external link. The field ‘Target Name’ is the primary key. For different ‘Target Domain/URL/Category’ a new ‘Target Name’ has to be used.

<table>
<thead>
<tr>
<th>Information</th>
<th>Type</th>
<th>Required</th>
<th>Max Length</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target Name</td>
<td>String</td>
<td>Required</td>
<td>Max Length</td>
<td>Possible Values</td>
</tr>
<tr>
<td>Target Domain</td>
<td>String</td>
<td>Required</td>
<td>Max Length</td>
<td>Possible Values</td>
</tr>
<tr>
<td>Target URL</td>
<td>String</td>
<td>Required</td>
<td>Max Length</td>
<td>Possible Values</td>
</tr>
<tr>
<td>Target Category</td>
<td>String</td>
<td>Required</td>
<td>Max Length</td>
<td>Possible Values</td>
</tr>
</tbody>
</table>

Notation:

```html
<a href='http://www.eanalytics.de' target='_blank' onclick='return eat_async.push(['eat_VisitorForwarding', 'eAnalytics','www.eanalytics.de','index.php','Homepage']);' >eAnalytics Homepage</a>
```
4.16.18. Mail opened

With the event ‘Mail opened’ you can track whether subscriber open the newsletter which you send. For this it is necessary to integrate an image request in the HTML newsletter.

Notation:

```html
<img src='http://www.mydomain.de/1000/eat1.gif?a=eAnalyticsNoVisitRequest&h=19,:,:,:,#MAILING_OBJECT#,#MAILING_ID#,#MAILING_RECIPIENT#,:,:,:,:,:,:,:,:,:,:,:&g=9999&z=eat_1.0' width='1' height='1' alt=''>
```

<table>
<thead>
<tr>
<th>Information</th>
<th>Typ</th>
<th>Pflichtfeld</th>
<th>höchst Länge</th>
<th>Beispiel</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAILING_OBJECT</td>
<td>String</td>
<td>ja</td>
<td>250</td>
<td></td>
</tr>
<tr>
<td>MAILING_ID</td>
<td>String</td>
<td></td>
<td>250</td>
<td></td>
</tr>
<tr>
<td>MAILING_RECIPIENT</td>
<td>String</td>
<td>ja</td>
<td>250</td>
<td></td>
</tr>
</tbody>
</table>

4.16.19. Client specific events

eAnalytics supports client specific events. You can use them for your specific business questions without changing the JavaScript file.

4.17. Client specific parameter

Special information that is not covered within the described parameters can be defined in the parameter `eat_clientSpec`.

Every attribute has its own name. If multiple information should be transferred, the name and value pairs are separated by the delimiter ‘|;’:

```javascript
    eat_async.push(['eat_setClientSpecificEvent',
                      '<param1>|;<param1_value>|;<param2>|;<param2_value>|;...']);
```

E.g.

```javascript
    eat_async.push(['eat_setClientSpecificEvent',
                      'CouponID|;CP_1884254']);
```

This will only cause the information to be sent to the eAnalytics Tag Server. You will have to expand the eAnalytics Data Processing (or ask us) in order to use them.
4.18. Request the embedded JavaScript file

This JavaScript Code requests the embedded JavaScript file:

```javascript
var u=("https:" == document.location.protocol) ? "#HTTPS_PATHTOFIELD#" : "#HTTP_PATHTOFIELD#";
var d=document, g=d.createElement('script');
var s=d.getElementsByTagName('script')[0];
g.type='text/javascript'; g.defer=true; g.async=true;
g.src=u+'eat_v1_2_1.js';
s.parentNode.insertBefore(g,s);
```

Figure 4 - Requesting the embedded JS file

The placeholders ‘#HTTP_PATHTOFIELD#’ AND ‘#HTTPS_PATHTOFIELD#’ have to be replaced with the actual path of the eAnalytics Tag JavaScript file on your web server or the URL to the JavaScript file on the eAnalytics server. Even if it is not necessary, it might be very useful to add an anchor (’#DATE#’)- containing the current date (generated in the web programming language you chose) - to the file request. This will force the browser to re-get the JavaScript file and along with it possible changes once a day.

Example:

```javascript
g.src=u+'eat_v1_2_1.js#20131010';
```

In case you prefer not to use the anchor, the code will look like this:

```javascript
g.src=u+'eat_v1_2_1.js';
```
5. CODE-BLOCK 3 – REQUEST

The CODE-BLOCK 3 sets a stop timer for the page load time and actually triggers the eAnalytics Tag Request:

```javascript
<script language="JavaScript" type="text/javascript">
    eat_async.push(['eat_setPageLoadTimeEnd', new Date()]);
    eat_async.push(['eat_featC']);
</script>
```
6. CODE-BLOCK 4 – NO JAVASCRIPT

The eAnalytics tag calls a JavaScript function to generate the requests. This only works if JavaScript is supported on the visitors’ browsers. If this is not the case and the JavaScript is deactivated than you can use the following code block which generates a basic tag request in the ‘no script’ part:

```html
<noscript>
    <img src="http://#TAGSERVERHOST#/1000/eat1.gif?a=#SESSIONID#&g=#DOMAINID#&e=#PAGETITLE#&f=#TOPIC1#%7C;#TOPIC2#%7C;#TOPIC3#&u=0" width="1" height="1" alt="" />
</noscript>
```

The placeholder `#TAGSERVERHOST#` has to be replaced by the name or IP address of the tag server (e.g. ‘eat.my-domain.de’) – usually the server you installed eAnalytics on. On SSL secured web pages the protocol has to be changed from ‘http://’ to ‘https://’.

```html
<noscript>
    <img src='http://www.mytagserver.de/1000/eat1.gif?a=342345jh2341jh52qwr2344fsadf&g=999&e=Produktansicht%20Hose%201234&f=Katalogseiten%7C;Produktansichten%7C;Produkt%201234&u=0'>
</noscript>
```

The following parameters can be defined and separated by ‘&’:

- SessionID (a)
- DomainID (g)
- Page Title (e)
- Topic of page (f)
- JavaScript deactivated (u)

Please note that the Page Title and Topic have to be URL-encoded. Business events can also be defined in this code block. eAnalytics provides support in such cases.

6.1. Session – ID (a)

Set your session ID to this parameter. In case you could not set this parameter you can leave it empty. In this case the ID is built based on the IP address.
e.g.: a=342345jh2341jh52qwr2344fsadf
6.2. Domain – ID (g)

This parameter sets the domain of the web page:

  e.g.: g=100

6.3. Pagetitle (e)

This parameter sets the title of the web page

  e.g.: e=Homepage

The set value is shown in the reports as the name of the pages.

6.4. Page topics (f)

This parameter sets the topics of a web page. Each of these three parameters can contain a string with a maximum of 250 characters. For this strings ‘umlauts’ and special characters have to be URL-encoded. Each topic is delimited by this string %7C;

  e.g.: f=Shop%20do%20it%20yourself%7C;electronic%20products%7C;cutout

6.5. JavaScript deactivated (u)

This parameter should not be changed. In code block 4 it is always set to ‘u=0’ in order to indicate that the request is generated by the no-script part and that JavaScript was not supported.

6.6. Anonymous IP and no IP in NoScript

If you implement a request with ‘eat0.gif’ the IP address will not be sent.

  eat1.gif = anonymous IP address will be sent
  eat0.gif = no IP address will be sent

```html
<noscript>
  <img src="http://#TAGSERVERHOST#/1000/eat0.gif?a=#SESSIONID&#g=#DOMAINID&#e=#PAGETITLE&#f=#TOPIC1%7C#TOPIC2%7C#TOPIC3#&u=0"
       width="1" height="1" alt="" />
</noscript>
```
7. TRACKING OBJECTION

eAnalytics offers a mechanism for tracking objection.

The following feature deactivates the tracking of the eAnalytics Tag:

```html
<script language='JavaScript' type='text/javascript'>
  eat_async.push(['eat_doNotTrack']);
</script>
```

The function should be implemented in a button, like ‘Do not track me’ or text link, on the web page so the visitor has the choice to get tracked or not.

A cookie ‘eat_DoNotTrack’ is set at the visitor computer. If the cookie exists, the eAnalytics Tag sends no request to tag server.

If a visitor does change his mind, there is a second JavaScript function that turns the tracking on again.

```html
<script language='JavaScript' type='text/javascript'>
  eat_async.push(['eat_doTrack']);
</script>
```

This deletes the cookie ‘eat_DoNotTrack’.